By – Dhananjay Trivedi



**Table of Contents**

1. Introduction
2. Architecture Overview
3. ProblemStatement

(A) [Objective](#_bookmark4)

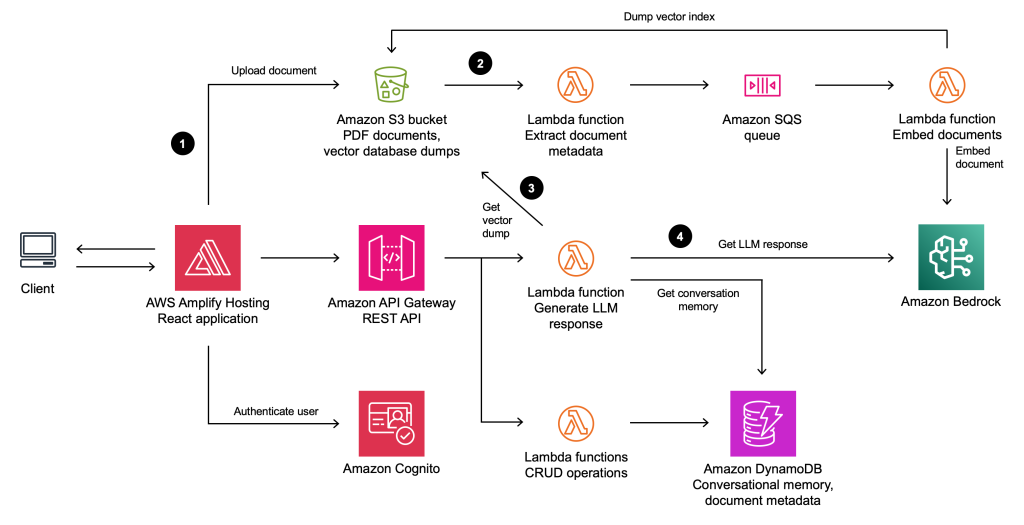
* + (B) [Abstractoftheproject](#_bookmark5)
  + (C) [Technologyused](#_bookmark6)

1. Implementation Steps
   * (A) Create Two S3 Buckets
   * (B) Set Up IAM Roles and Policies
   * (C) Develop AWS Lambda Function
2. Testing the Application
3. Appendix

## ****Introduction****

A serverless image processing application can be built using AWS services like S3, Lambda, and the Sharp Node.js library. When an image is uploaded to an S3 bucket, it triggers a Lambda function that automatically resizes and optimizes the image using Sharp. The processed image is then stored in another S3 bucket or folder. This approach provides automatic scalability, reduced costs, and high-speed image manipulation without compromising quality. The solution also manages version control and supports flexible storage and compute, making it ideal for websites and mobile apps needing efficient media handling.

1. **Architecture Overview**

****

## ****Problem Statement****

### ****(A). Objective****

The main objective of this project is to develop a **serverless** and **scalable image processing application** that automatically resizes and optimizes images upon being uploaded to an Amazon S3 bucket. This will reduce manual intervention, improve web performance, and decrease storage and bandwidth usage.

### ****(B). Abstract of the Project****

The proposed application leverages **AWS Lambda** functions triggered by **S3 upload events**. When an image is uploaded to a source S3 bucket, the Lambda function is automatically triggered, processing the image (e.g., resizing and compression), and storing the optimized output into a destination S3 bucket. The system uses serverless architecture to ensure cost-efficiency, scalability, and minimal maintenance overhead.

This application is ideal for dynamic websites, mobile apps, and media platforms needing fast and automatic media optimization.

### ****(C). Technology Used****

* **AMAZON WEB SERVICE**

1. **Implementation Steps**

## LAB STEPS:-

Step 1: Sign in to AWS Management Console

1. Click on the Open Console button, and you will get redirected to AWS Console in a new browser tab.

2. On the AWS sign-in page,

•Leave the Account ID as default. Never edit/remove the 12 digit Account ID present in the AWS Console. otherwise, you cannot proceed with the lab.

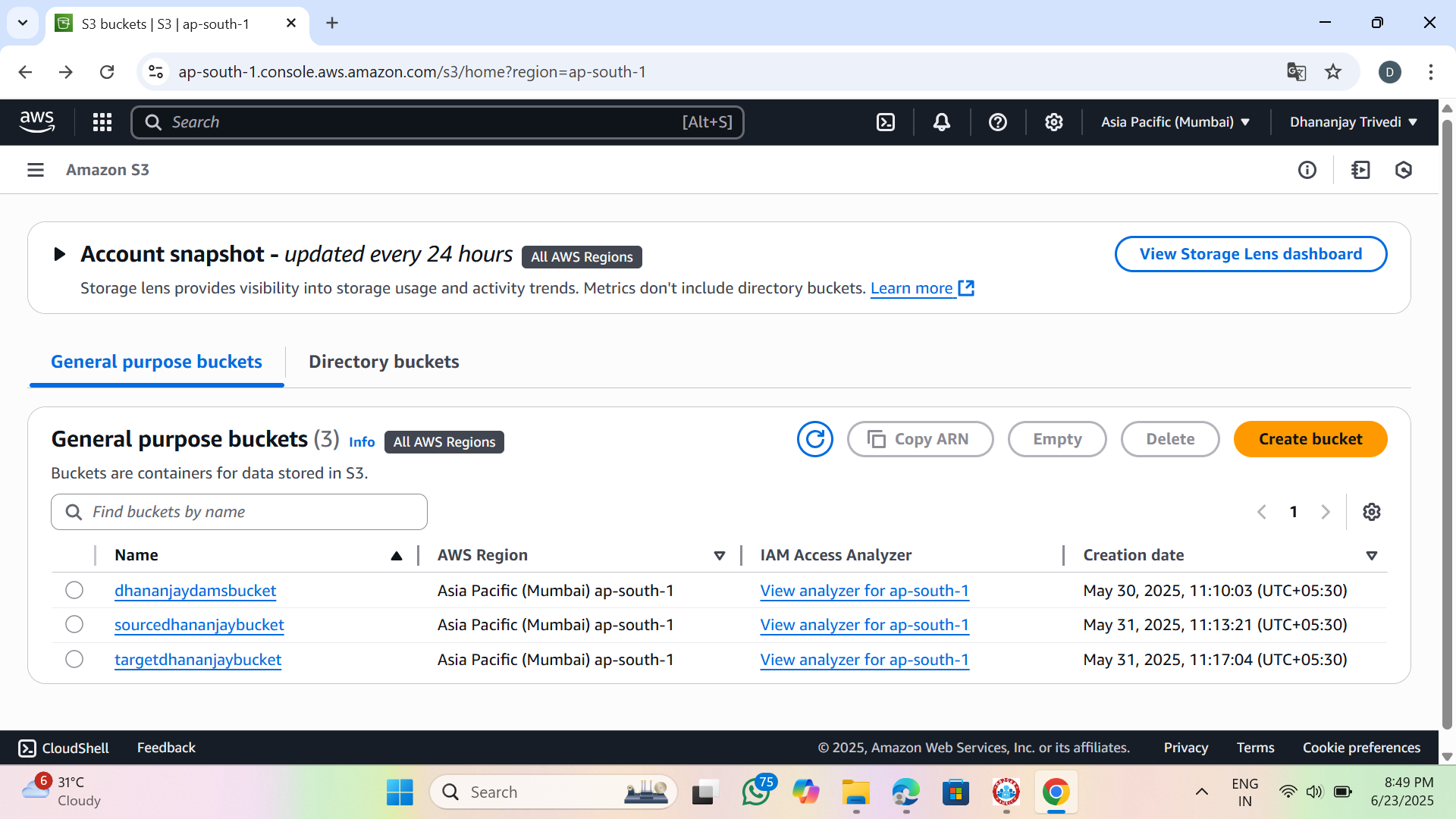
• Now copy your User Name and Password in the Lab Console to the IAM Username and Password in AWS Console and click on the Sign in button.

3. Once Signed In to the AWS Management Console, Make the default AWS Region as AP (Pacific Mumbai) ap-south-1.

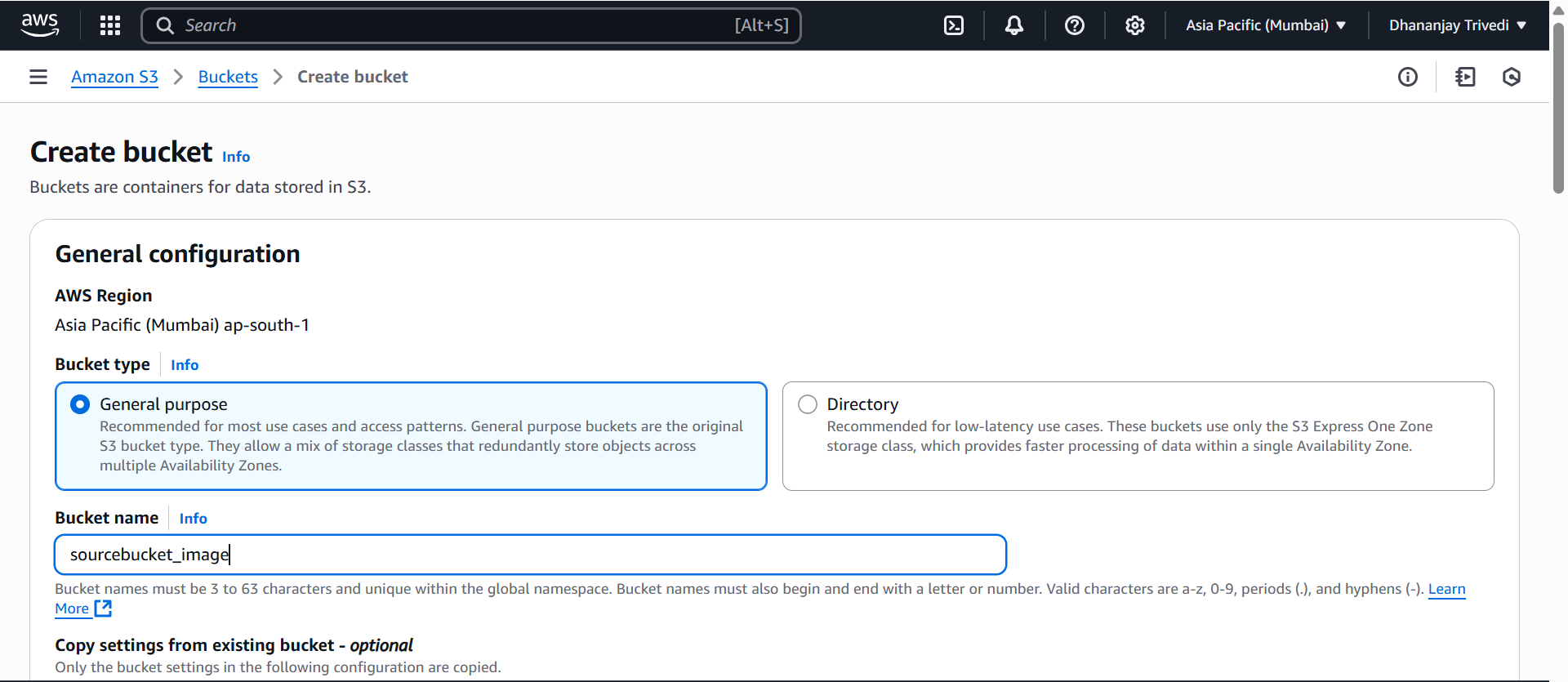
**4.1 Create Two Amazon S3 Buckets**

In this task, we will create two AWS S3 buckets i.e the source bucket and the destination bucket by providing the required configurations like name, region etc.

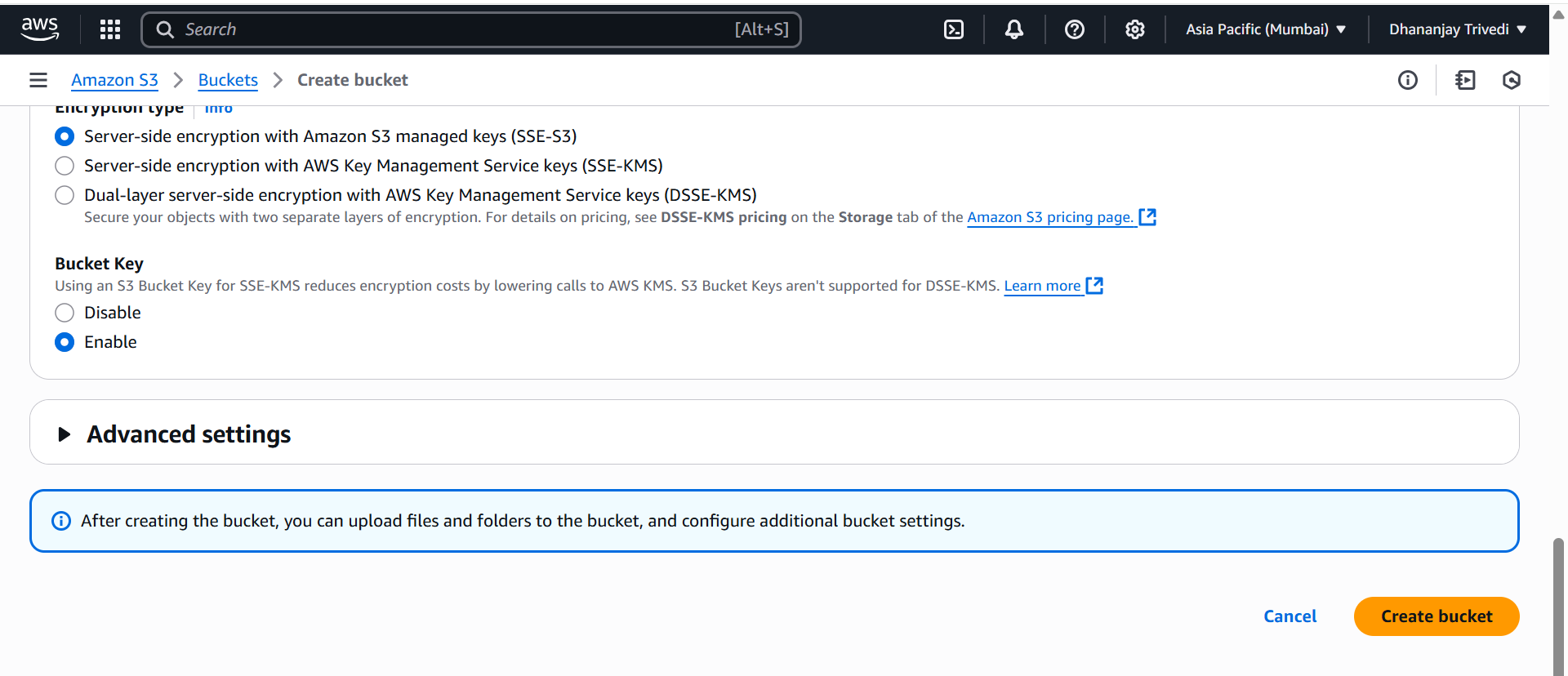
1. Navigate to the Services menu in the Top, then click on S3 in the storage section



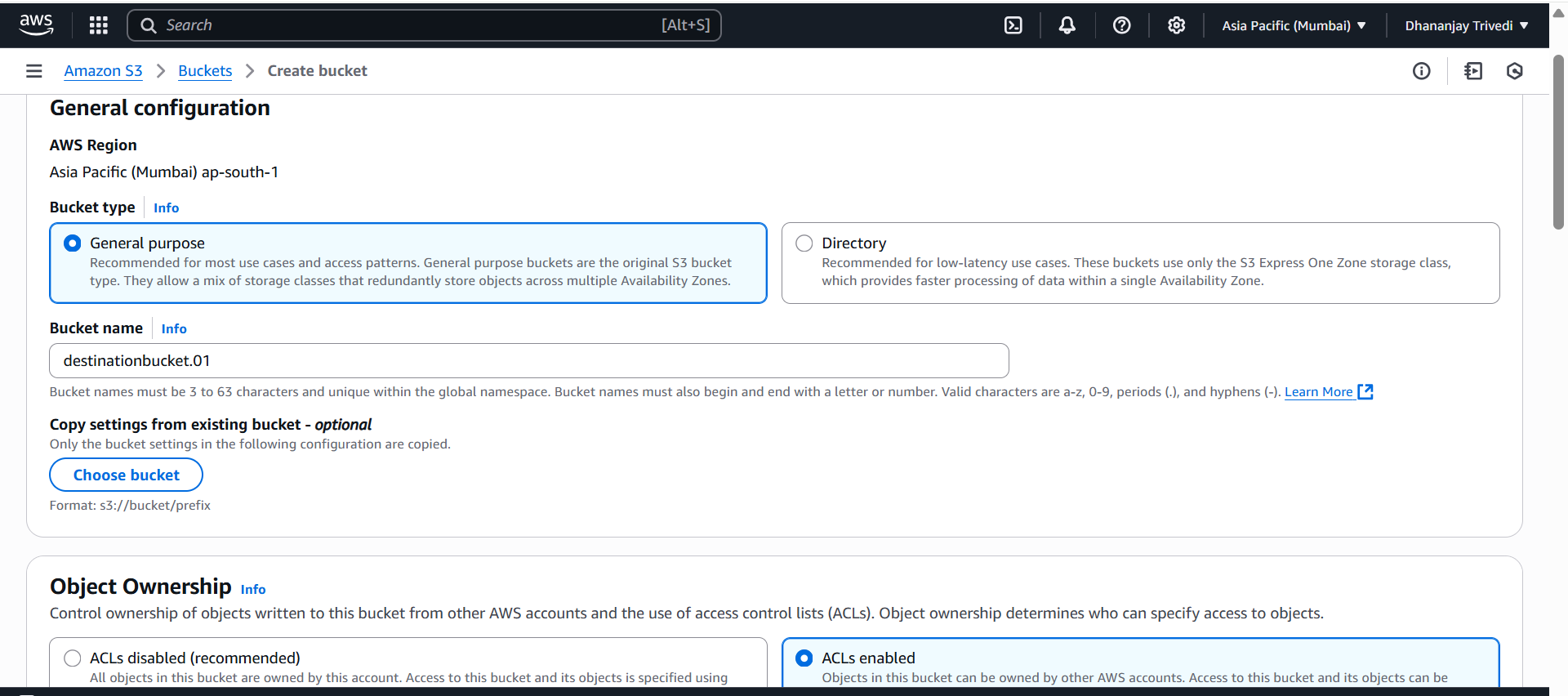
* .Click on Create Bucket button.
* .**Create Source Bucket**



* Block public Access or click the I acknowledge button then ,Leave Other settings as Default and click on the **Create Bucket** button



* Once the Bucket is created successfully, Select your S3 bucket.
* Click on the Copy ARN button to copy the ARN.
* Save the source bucket ARN in a text file for later use.
* **Create Destination Bucket**



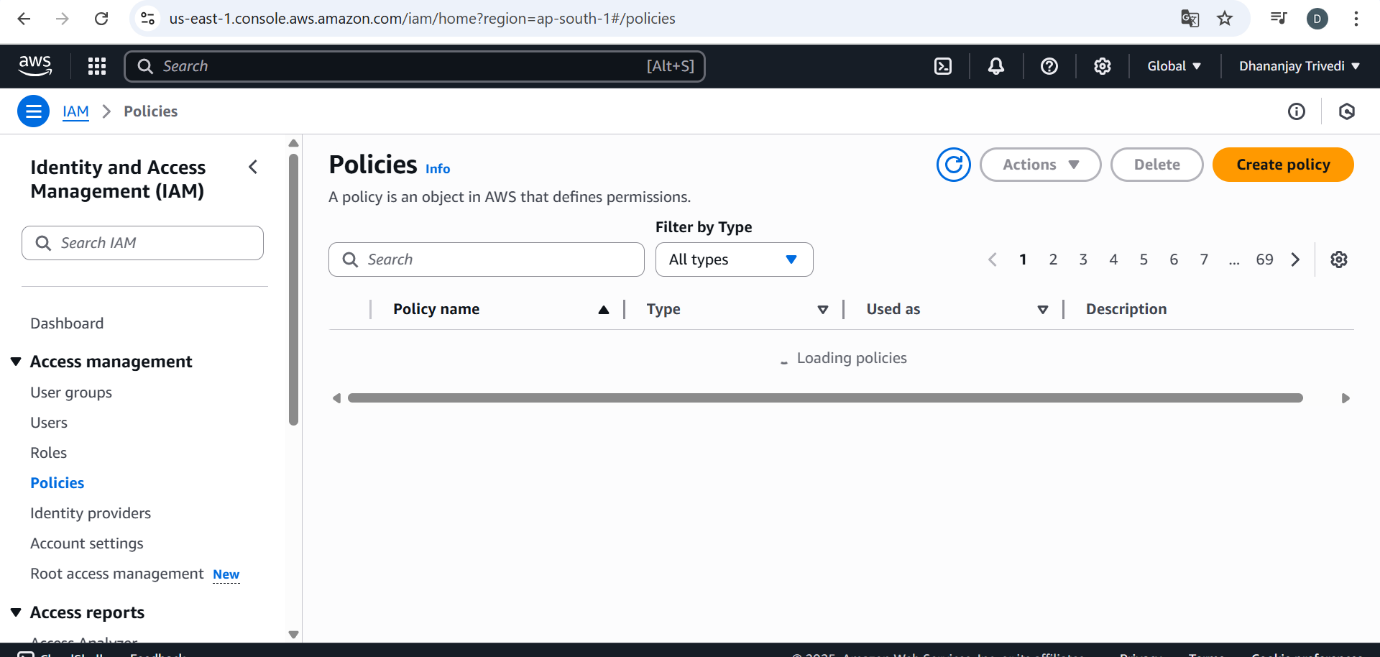
>Do All Process Same as Source Bucket Everything

* Save the destination bucket ARN in a text file for later use.

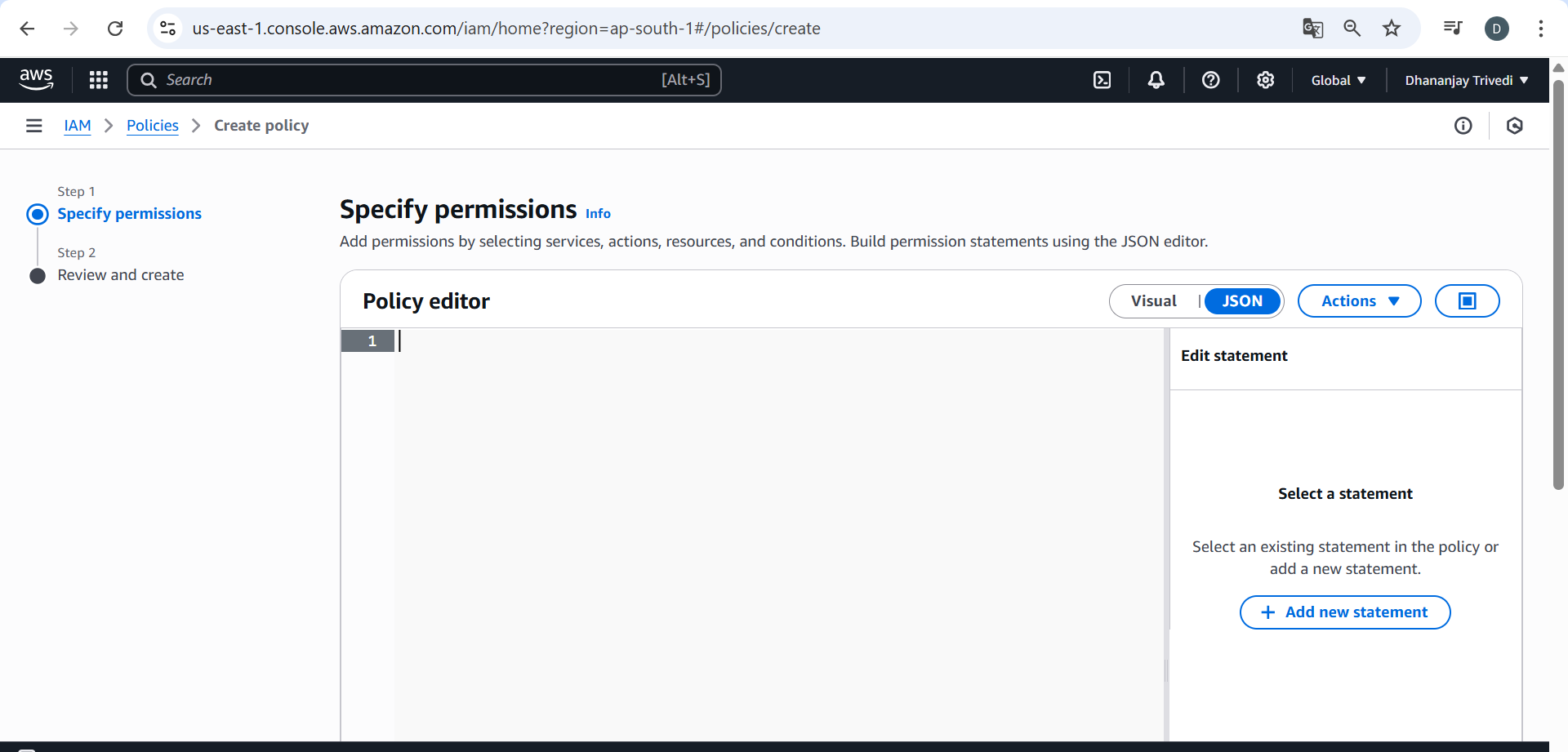
Important when you create bucket then Upload any **Image.jpg** in the source bucket

**4.2 Set Up Policies and IAM Roles**

* Go to **Services** and Select **IAM** under **Security, Identity and Compliance.**
* Click on **Policies** in the left navigation bar and click on the **Create policy** button.

****

* Click on the **JSON** tab, Remove the existing code and copy-paste the below policy statement into the editor:

****

•Policy JSON:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow", "Action": [ "logs:PutLogEvents",

"logs:CreateLogGroup", "logs:CreateLogStream"

],

"Resource": "arn:aws:logs:\*:\*:\*"

},

{

"Effect": "Allow",

"Action": ["s3:GetObject"],

"Resource": "source bucket arn paste " /\*"

},

{

"Effect": "Allow", "Action": ["s3:PutObject"],

"Resource": “destination bucket arn paste” /\*"

}

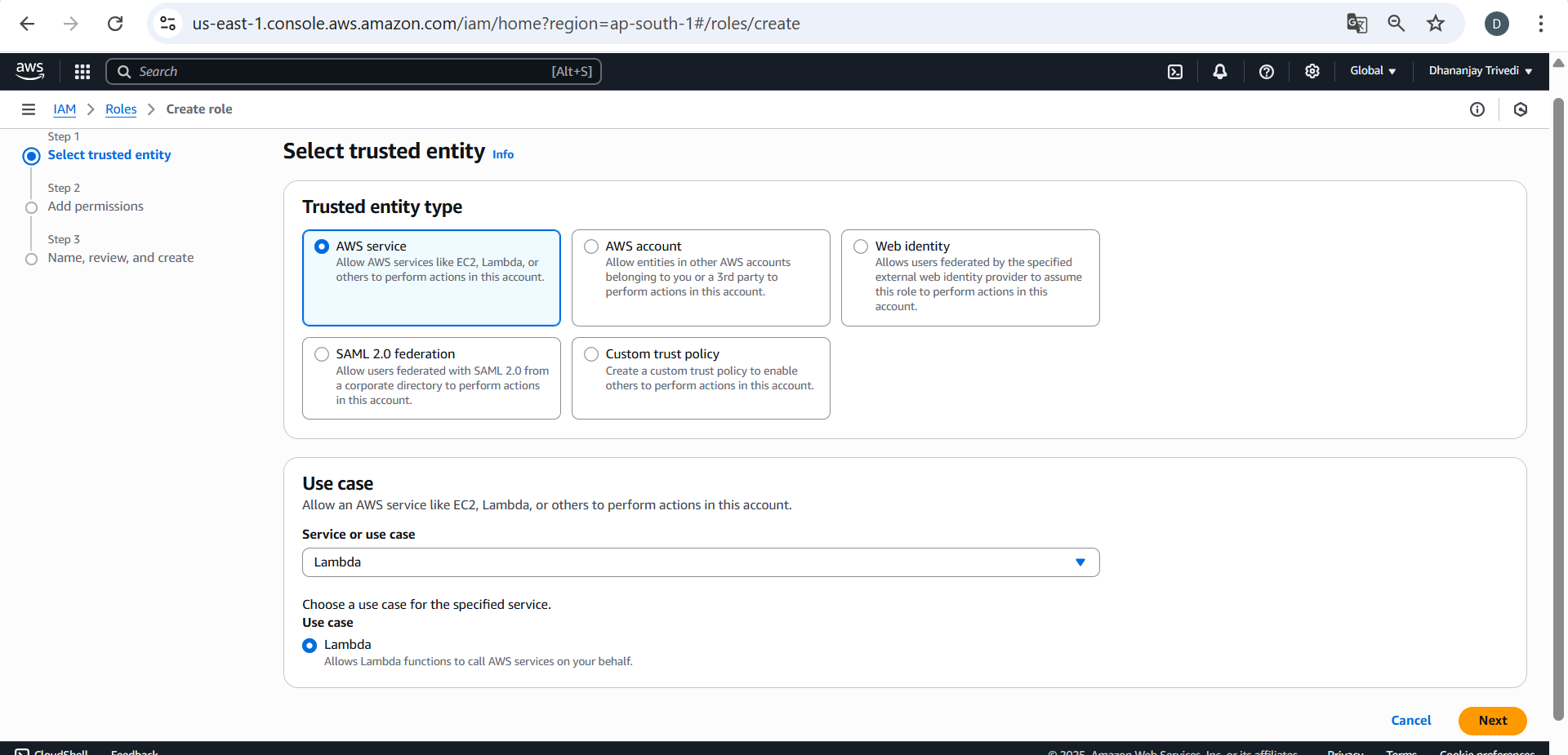
]

}

* Leave Everything as default and click on **Next** button.
* On the Review Policy page:
* Enter **Policy Name and** Click on the **Create policy** button

# Create an IAM Role

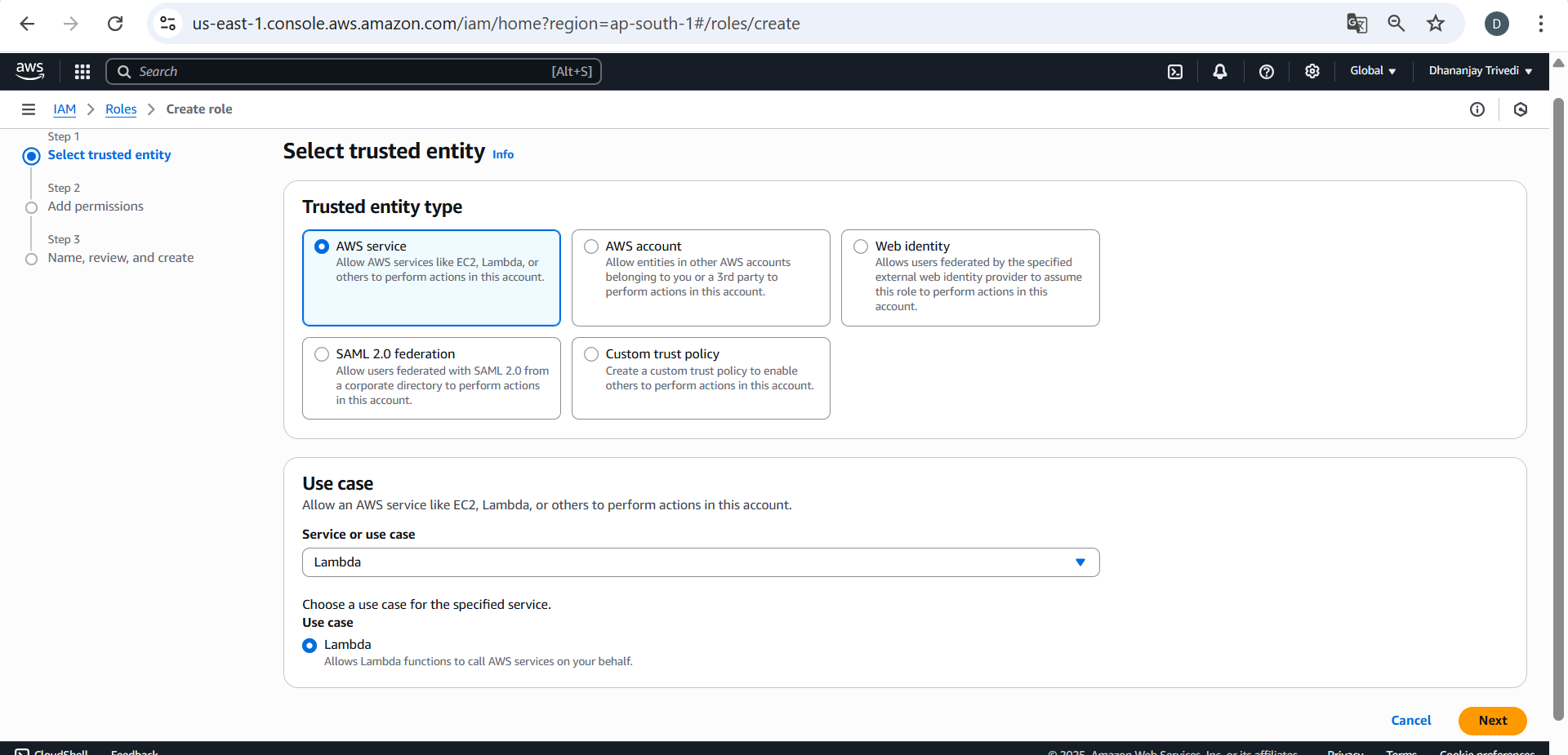
* .In the left menu, click on **Roles** and click on the **Create Role** button then



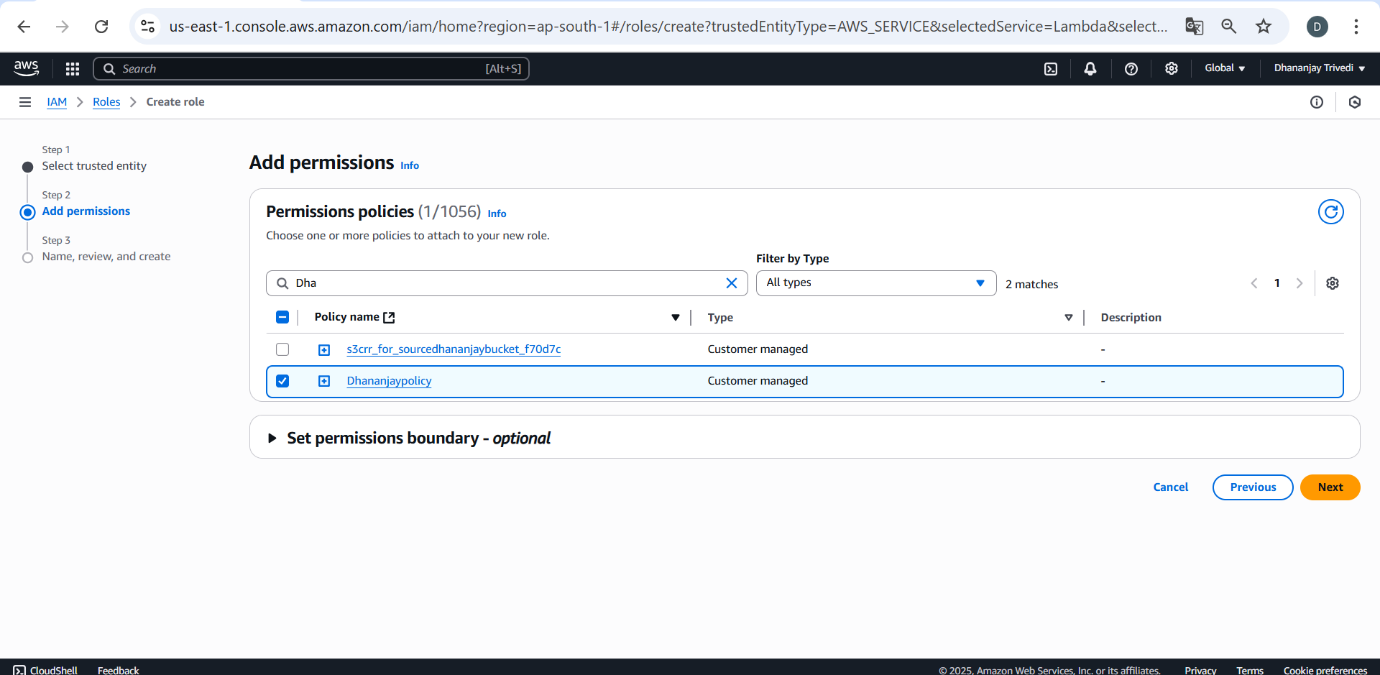
* . Select Lambda from AWS Services list.

•From Trusted Entity Type: Select **AWS Service** • From Use case: **Select Lambda**

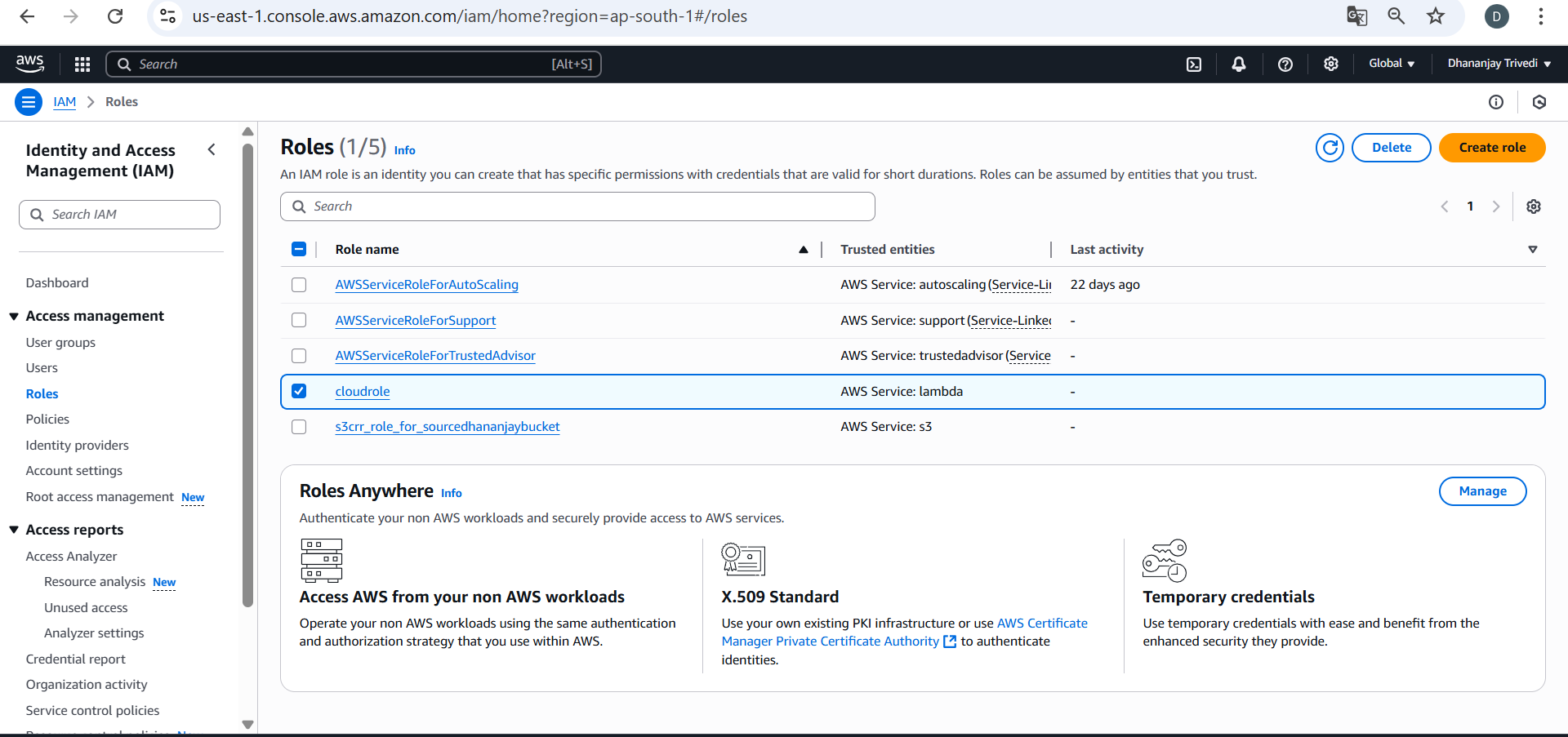
• Click on Next button.



* Select your **policy** and click on the **Next** button.

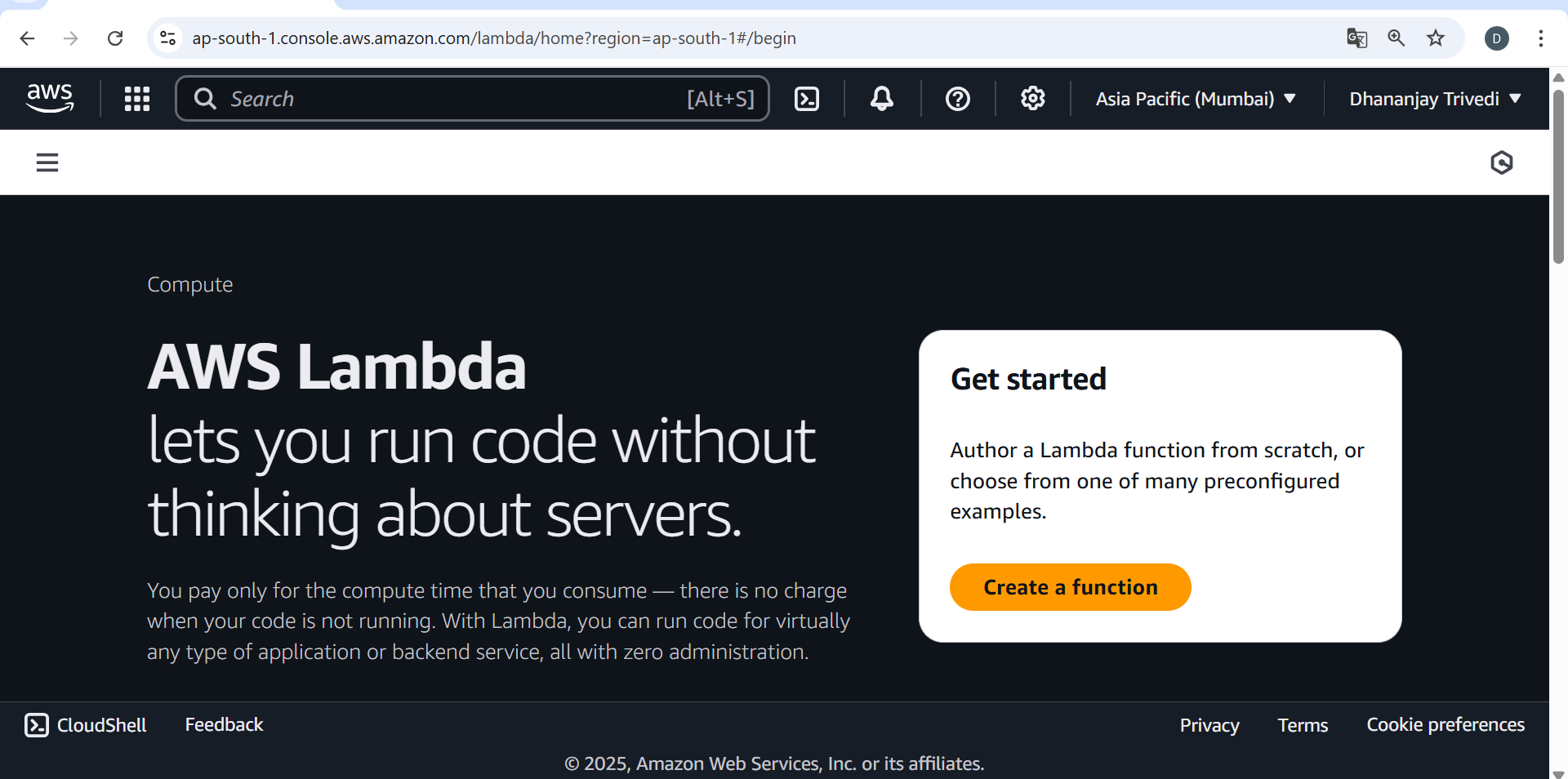


* **Role Name**: Enter **imagerole**
* Click on the **Create Role** button.
  + You have successfully created an IAM role by name cloudrole.

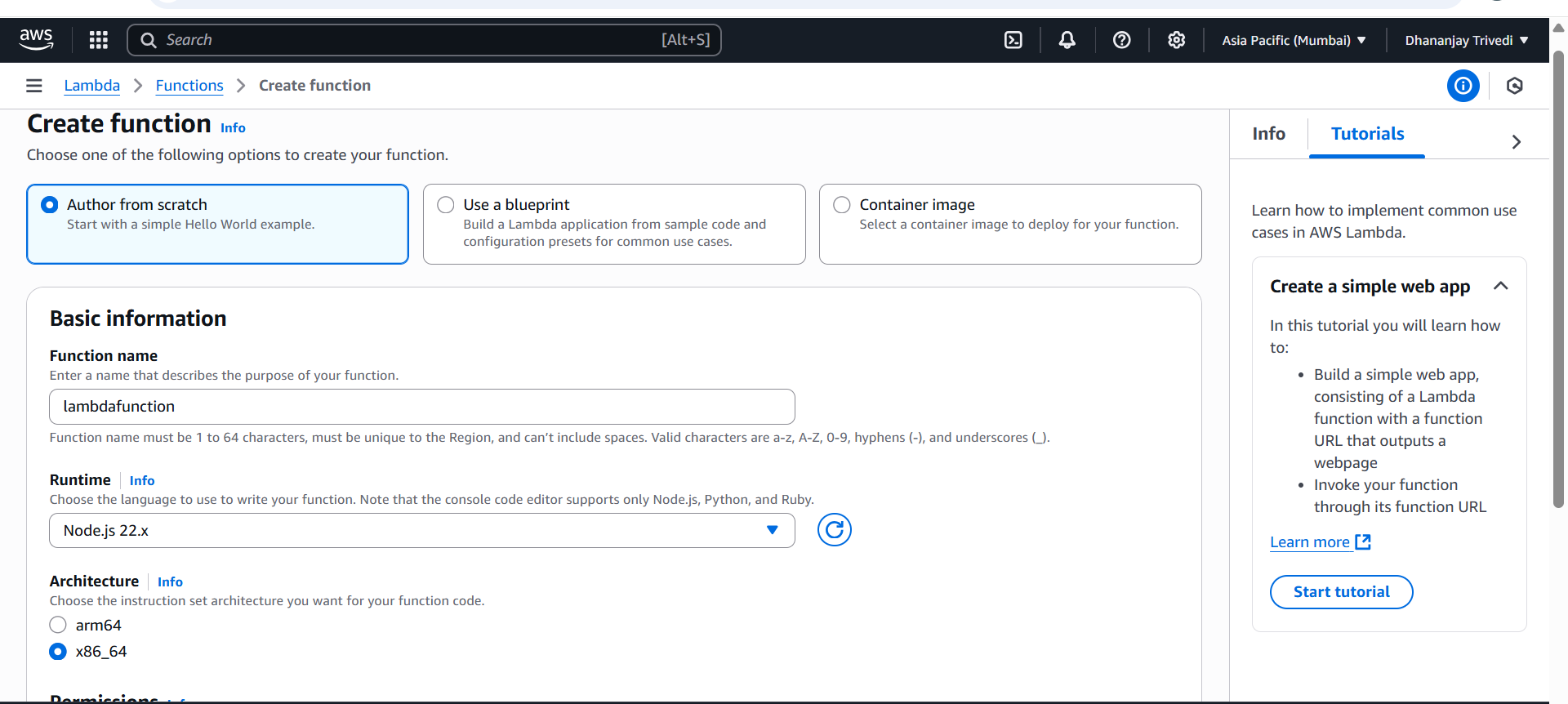


# Creating Lambda function

* Go to AWS Lambda Console, Navigate to functions section . Click **Create function**

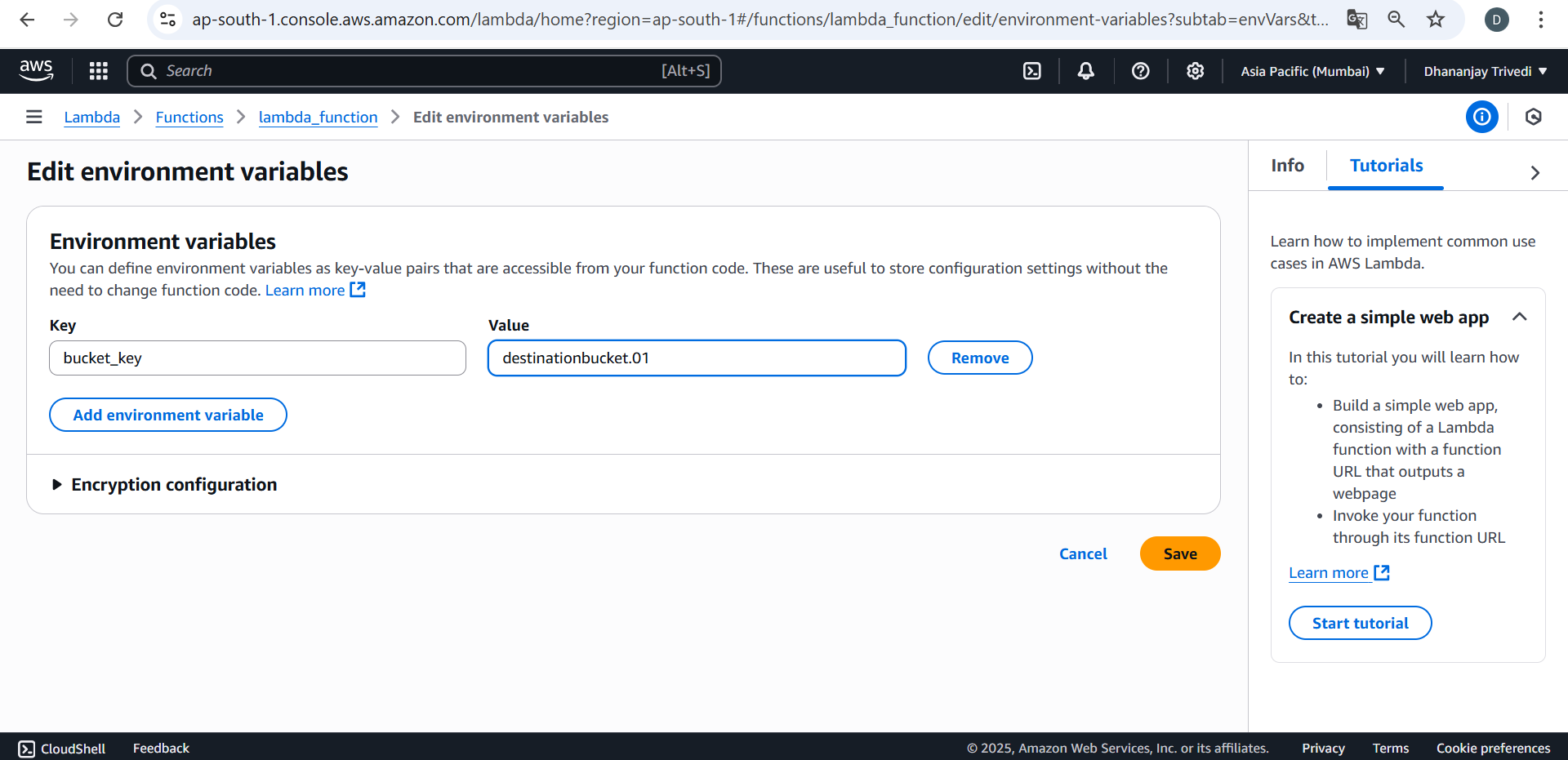


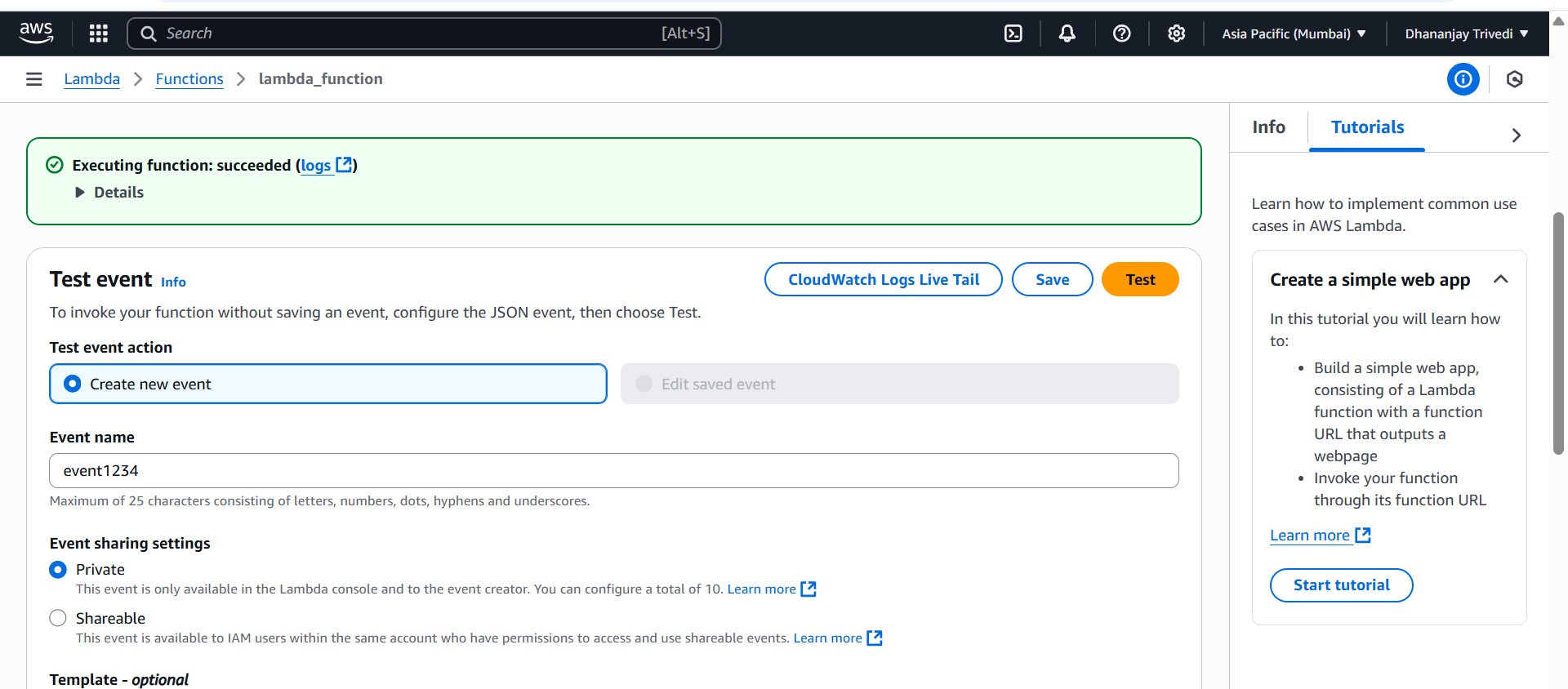
* Give lambda Name and select runtime and Leave all other settings as default.
* Change Default execution role.
* And click use existing role 🡪 select your role name.
* Click 🡪Create function



* Go to configuration 🡪 Select🡪 Environment Variables click **Edit 🡪 give the key & value name**

Key-> DEST\_BUCKET & value-> destination bucket ‘Name’



* **Test Lambda Function**
* Go to AWS Lambda console. Navigate to Functions section.
* open function then will be created
* open test console & create event or keep it default
* template🡪 s3-put
* **Also you see a default code same as this**
* **EVENT JSON:**

**{**

**"Records": [**

**{**

**"eventVersion": "2.0",**

**"eventSource": "aws:s3",**

**"awsRegion": "ap-south-1", #changethis**

**"eventTime": "1970-01-01T00:00:00.000Z",**

**"eventName": "ObjectCreated:Put",**

**"userIdentity": {**

**"principalId": "EXAMPLE"**

**},**

**"requestParameters": {**

**"sourceIPAddress": "127.0.0.1"**

**},**

**"responseElements": {**

**"x-amz-request-id": "EXAMPLE123456789",**

**"x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"**

**},**

**"s3": {**

**"s3SchemaVersion": "1.0",**

**"configurationId": "testConfigRule",**

**"bucket": {**

**"name": " Example source bucket name ", #changethis**

**"ownerIdentity": {**

**"principalId": "EXAMPLE"**

**},**

**"arn": "arn:aws:s3:::Example source bucket name" #changethis**

**},**

**"object": {**

**"key": "pexels-pixabay-47547.jpg",**

**"size": 1024,**

**"eTag": " test%2Fkey ", #changethis**

**"sequencer": "0A1B2C3D4E5F678901"**

**}**

**}**

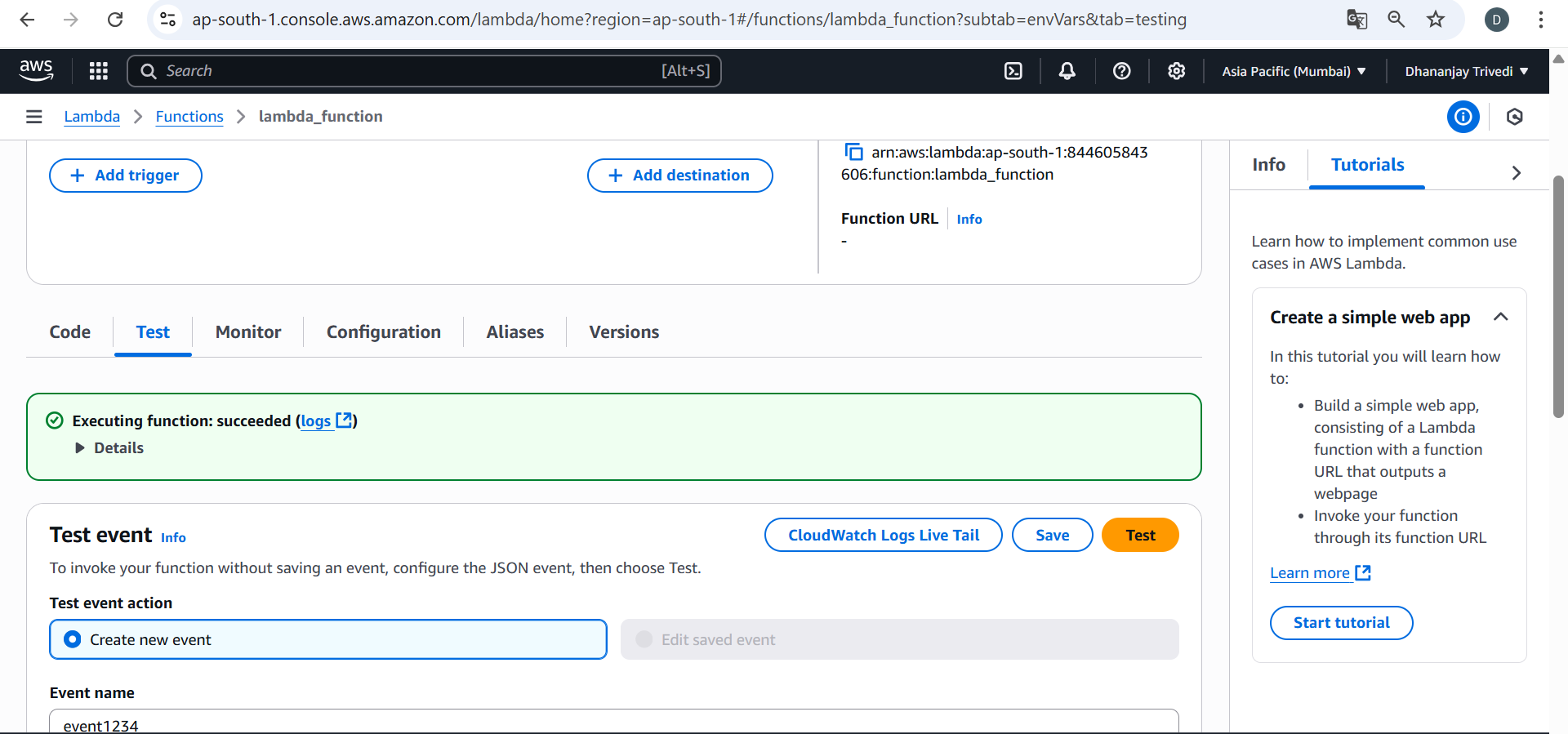
**}**

**]**

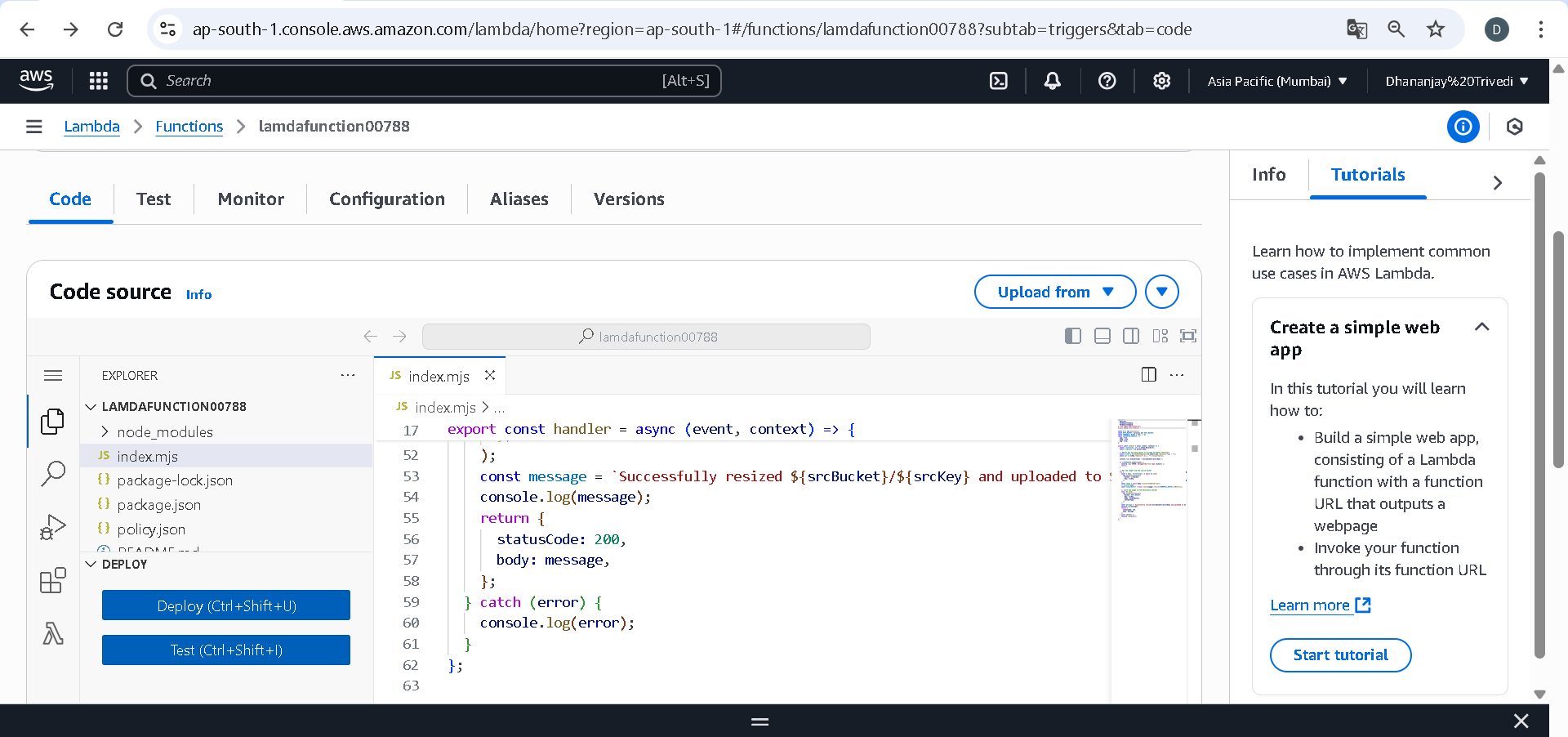
**}**

**– Change the example bucket to the source bucket and also change "test%2Fkey", to your uploaded image name and must change your aws region name like ‘ap-south-1’, know it’s ready to see so click on the test button**

* **Now click to save & Test** 🡪**see the Execution func succeeded**

****

* Now go to the **CODE** section & Upload the **ZIP**. File -> [function.zip](file:///C:\Users\DELL\Downloads\function.zip)



* And Here we go  **test** the code you can see this output

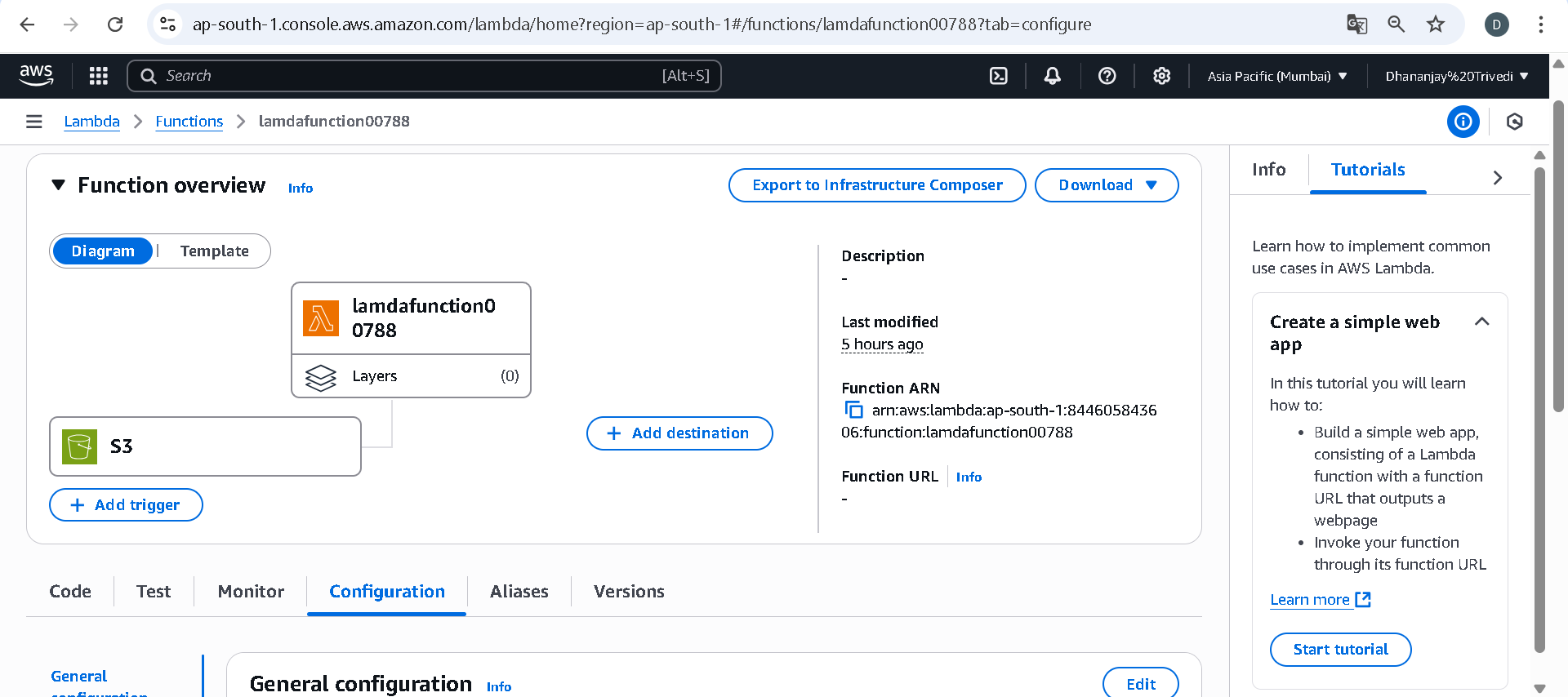
**"statusCode": 200,**

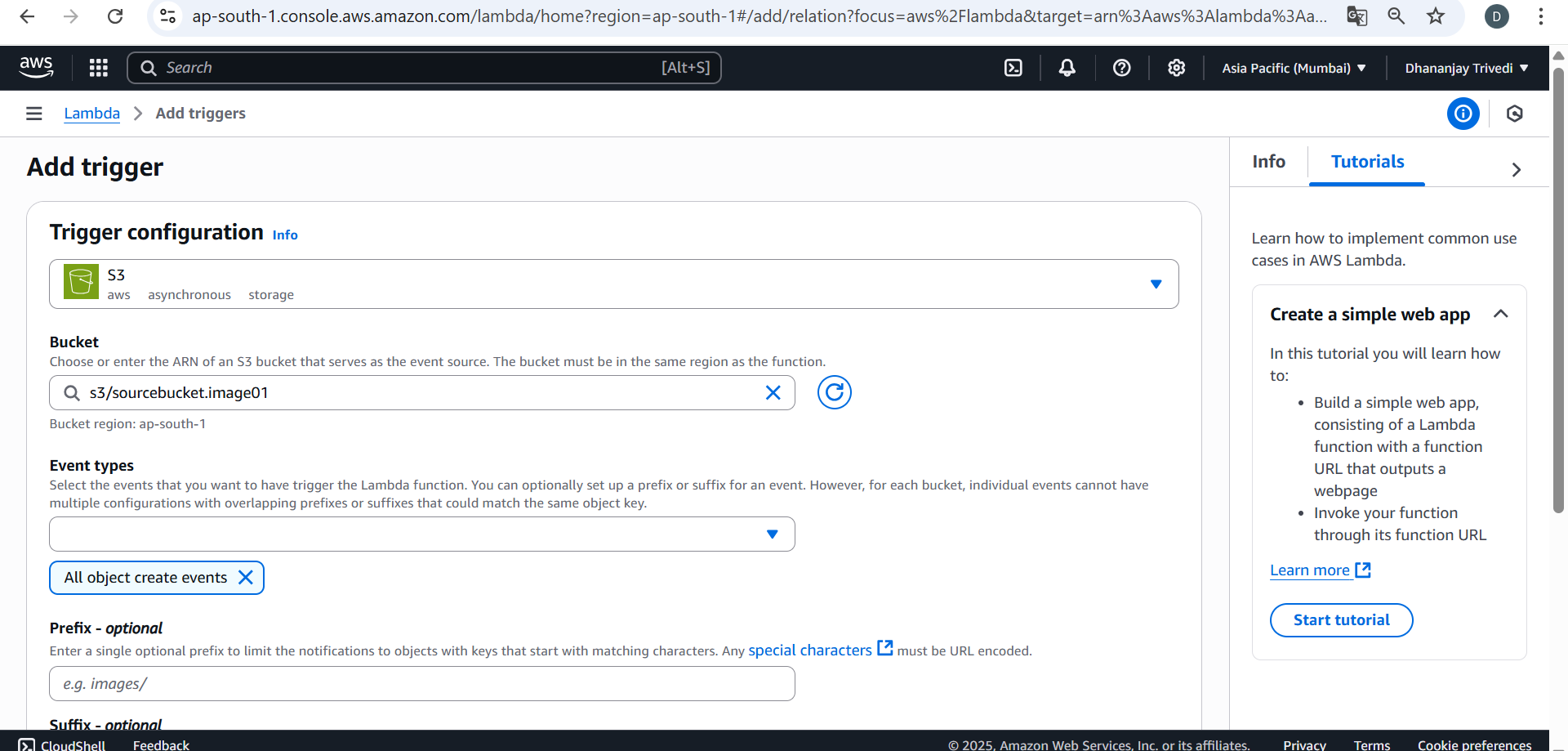
**"statusCode": 200,**

**"body": "Successfully resized souc-bucket001/pexels-pixabay-47547.jpg and uploaded to de-bucket001/pexels-pixabay-47547.jpg"**

**"body": "Successfully resized souc-bucket001/pexels-pixabay-47547.jpg and uploaded to de-bucket001/pexels-pixabay-47547.jpg"**

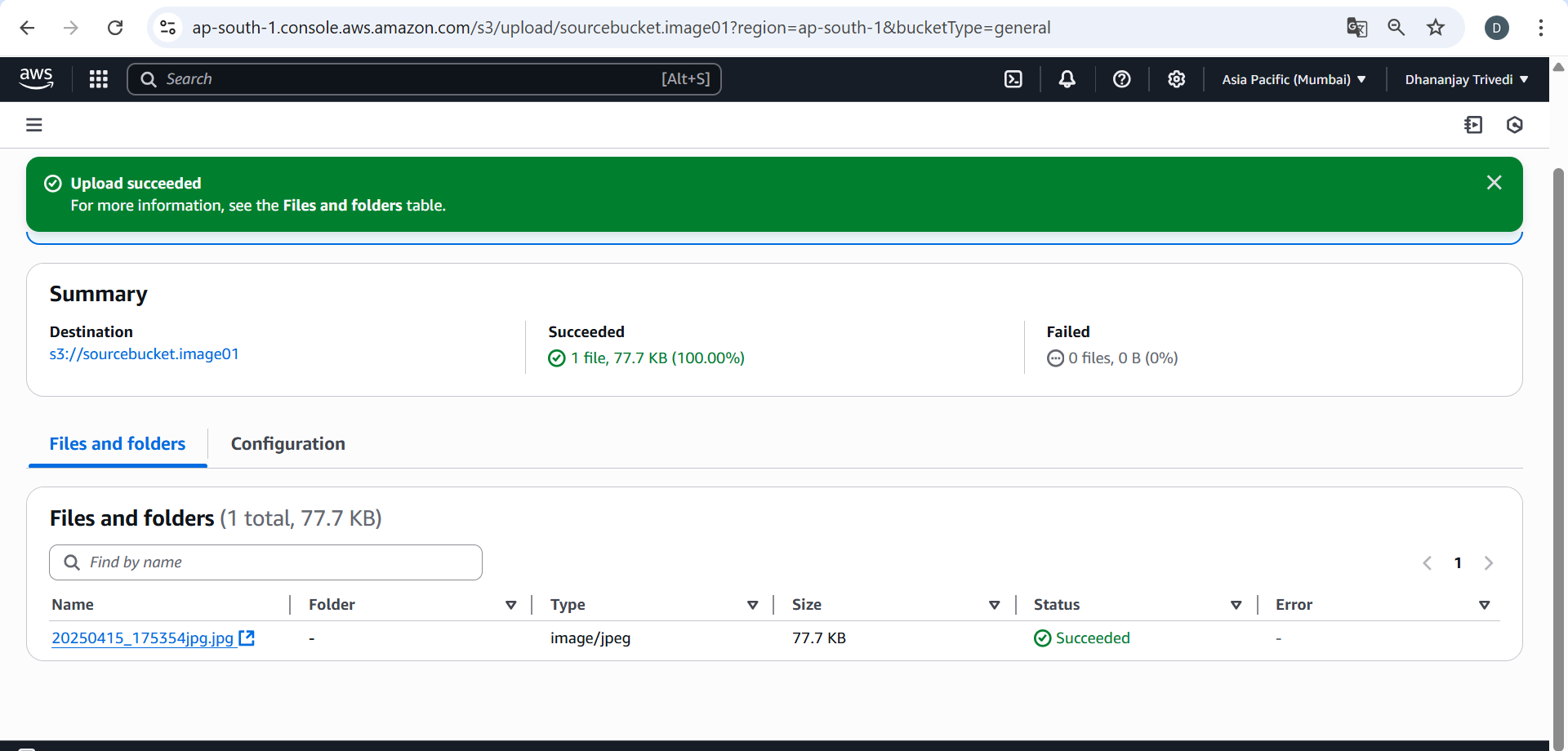
* **Creating S3 Trigger in the Lambda**
* Add trigger
* Select s3
* choose source Bucket name
* Now Add





1. **Testing the Application**

* **Uploaded image in Source Bucket**

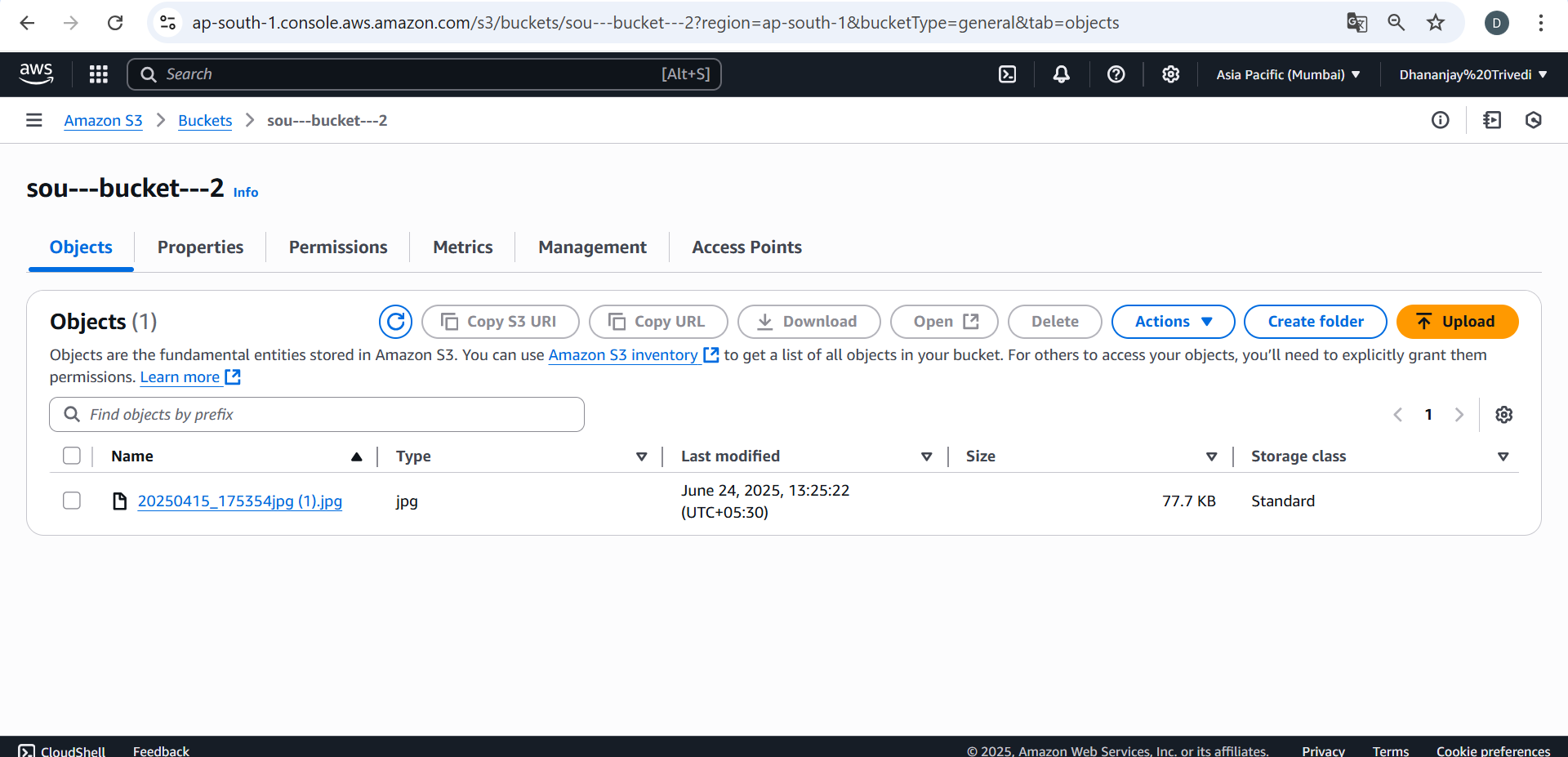
****

* **Original Image** 🡪

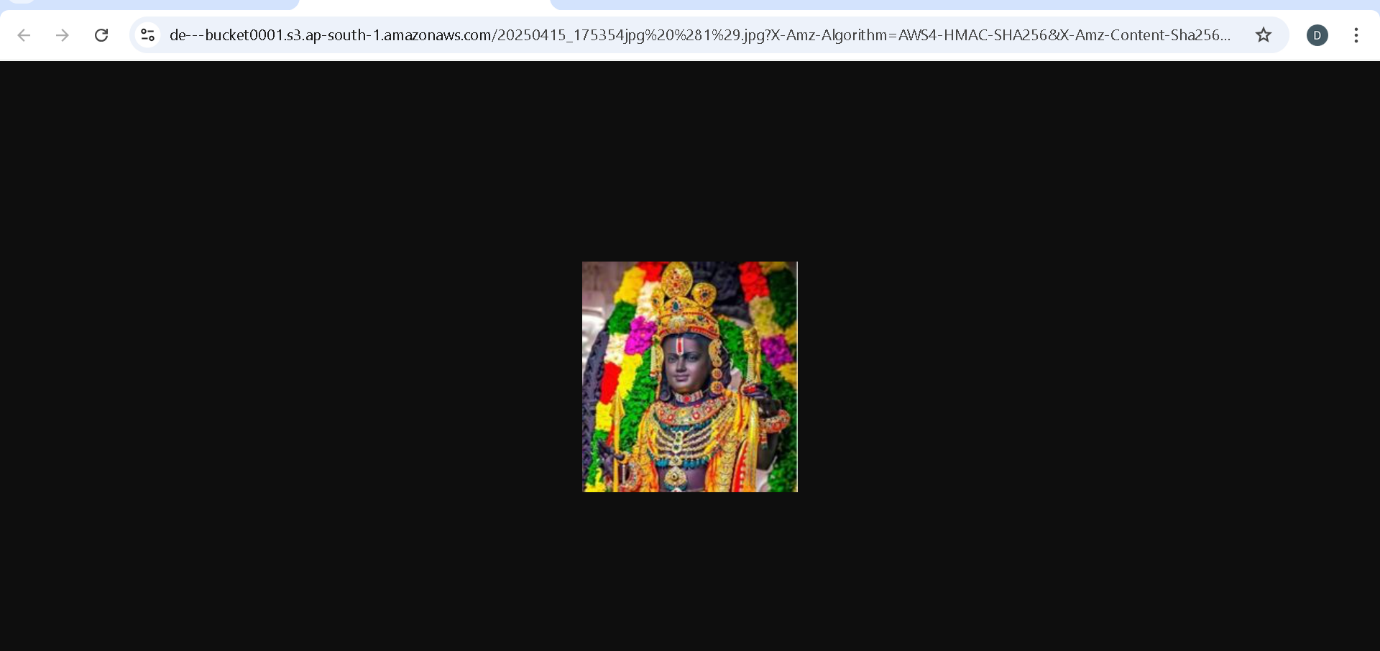


* Destination Bucket 🡪 Automatically Fetch the uploaded image

And it will r esize it .

0

* **Resize Image** 🡪

****